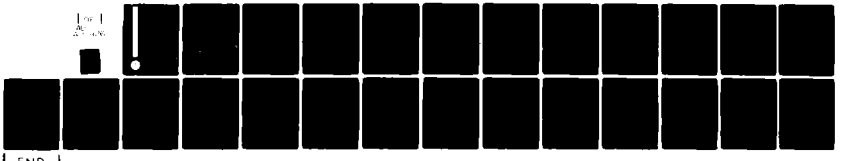
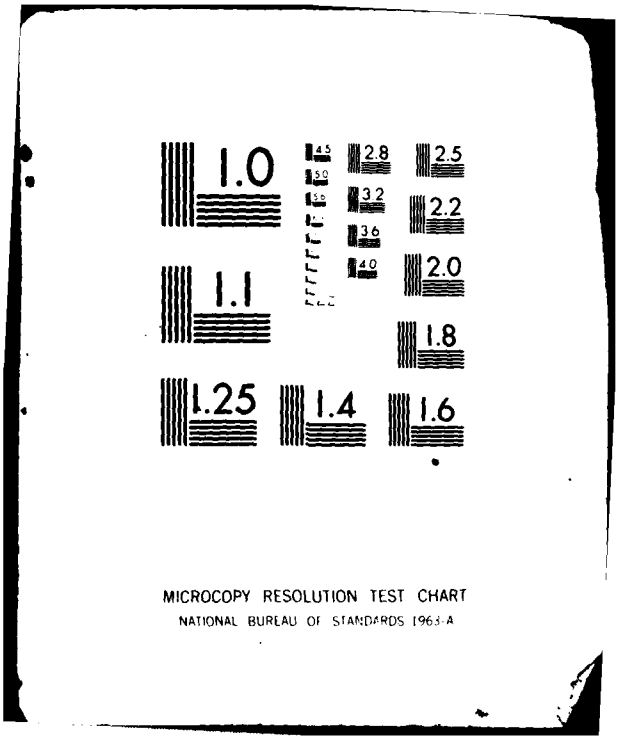
AD-A111 926 STATE UNIV OF NEW YORK AT BUFFALO AMHERST GROUP FOR --ETC F/6 9/2
THE ADVICE-TAKER/INQUIRER.(U)
DEC 81 G L SICHERMAN AFOSR-81-0220
UNCLASSIFIED TR-193 AFOSR-TR-82-0094 NL



END
DATE FILMED
4-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

THE ADVICE-TAKER/INQUIRER*,**

BY

GEORGE L. SICHERMAN
GROUP FOR COMPUTER STUDIES OF STRATEGIES

DECEMBER 1981

DEPARTMENTAL TECHNICAL REPORT #193
GCSS TECHNICAL REPORT #5

*THE WORK DESCRIBED HEREIN IS SUPPORTED BY AIR FORCE OFFICE OF SCIENTIFIC RESEARCH AFOSR 81-0220.

**THIS REPORT IS BASED ON THE AUTHOR'S PH.D. DISSERTATION PROPOSAL SUBMITTED TO THE FACULTY OF THE DEPARTMENT OF COMPUTER SCIENCE.

STATE UNIVERSITY OF NEW YORK AT BUFFALO

Department of Computer Science

(2)

THE ADVICE-TAKER/INQUIRER*,**

BY

GEORGE L. SICHERMAN
GROUP FOR COMPUTER STUDIES OF STRATEGIES

DECEMBER 1981

DEPARTMENTAL TECHNICAL REPORT #193
GCSS TECHNICAL REPORT #5

*THE WORK DESCRIBED HEREIN IS SUPPORTED BY AIR FORCE OFFICE OF SCIENTIFIC RESEARCH [redacted] 81-0220.
AFOSR

**THIS REPORT IS BASED ON THE AUTHOR'S PH.D. DISSERTATION PROPOSAL SUBMITTED TO THE FACULTY OF THE DEPARTMENT OF COMPUTER SCIENCE.

CONTENTS

1 Introduction

This report describes an intelligent system, the Advice-Taker/Inquirer, being developed at the State University of New York at Buffalo (SUNY/AB). Its principal capabilities are:

1. accept expert advice in the form of principles;
2. accept expert advice in the form of examples;
3. make inquiries to the advisor when the advice is vague, incomplete, or contradictory;
4. apply the advice in the field; and
5. evaluate and adapt its advice on the basis of experience.

The motivation for such a system is obvious. Many task environments are so complex that it is impossible or impractical to specify an effective algorithmic strategy for them. In such environments the best way to specify a strategy usually is in terms of principles. The value of being able to use examples is that a single example may be more instructive than a large number of principles. The system makes inquiries when it detects logical problems with the advice, in order that it may not blindly follow an illogical strategy. Finally, the system applies and evaluates the advice in order to detect practical problems with the advice. If the system has access to a system that simulates the problem domain, it can detect practical problems before applying the strategy in the real world. This capability is very desirable from a practical standpoint.

2 Background and related work

2.1 Automatic programming.

Since the earliest days of computing, programmers have recognized that the power of the computer might be applied to the very task of programming. Compilers were the first major achievement of this kind. (It is worth remembering that FORTRAN was envisioned as an alternative to programming.) Today programming languages are not regarded as "automatic programming" systems, the term being applied instead to systems that construct programs according to the user's statements about what he wants to accomplish. See [BIERMANN 1976] for a review of various automatic programming systems till 1975.

Most such systems are of little use in constructing programs for artificial intelligence (AI), because they are designed to construct algorithmic programs rather than heuristic ones. The only examples of Biermann's for AI are [SUMMERS 1975], which constructs LISP functions for transforming list structures by example, and Winograd's system [WINOGRAD 1972] for translating natural language into PLANNER.

2.2 The advice-taker.

The earliest proposal for a system that would accept heuristic input came from J. McCarthy [McCARTHY 1968]. It described a program called the Advice-Taker, essentially a theorem-prover, in which the laws of deduction would be stored in the same symbolism as the premises. The control structure would consist merely of a few "hard-programmed" heuristics for observing and matching patterns. He did not define these heuristics. The practical interest of the Advice-Taker lay in its ability to accept heuristics as input, then apply the

heuristics in processing the data. It was one degree more flexible than an ordinary program that could process various data sets but could not vary its processes.

2.3 The advice-taking chess player.

The first significant achievement in automatic advice-taking was Zobrist and Carlson's advice-taking chess player [ZOBRIST and CARLSON 1973]. This program accepted heuristics from a chess master. The communication language devised by the authors was opaque, but simple enough for the master to learn easily. Each heuristic described one kind of feature of a position and how to compute that feature's value. It was generally considered to play more like a human player than most other chess programs of its time.

2.4 Teiresias.

The best-known example of a knowledge-refinement system is Davis's TEIRESIAS [DAVIS 1979], a sophisticated agent for maintaining and debugging consultation systems such as MYCIN [SHORTLIFFE and BUCHANAN 1975]. TEIRESIAS can help the user track down bugs, identify anomalies that might be bugs, describe the inferential operation of its host program, and identify situations that are not specifically covered by the advice but perhaps ought to be.

This naturally requires a high order of generalization and pattern recognition. Davis's method was to give TEIRESIAS not only access to the consultation program and its knowledge base, but also a model of its own knowledge. To Davis a model was "a

compact, high-level description of structure, organization, or content that may be used both to provide a framework for lower-level processing, and to express expectations about the world." (p. 140) Davis designed his rule models with a view to their intended use. Each contained (1) examples of rules described by the model; (2) a list of more general models; (3) a list of more specific models; and (4) a description of the rules. The description would characterize a typical rule, the premise, the action, typical attributes involved, and correlations of these attributes. This structure provided an ideal high-level framework for describing a rule-based consultation system.

TEIRESIAS used natural language, but restricted its use as much as possible. The program prompted the user to give short answers whenever possible; long answers were not parsed but keyword-matched.

2.5 FOO.

The most advanced advice-taker to date is Mostow's FOO [MOSTOW 1981]. Mostow provided FOO with about 400 rules. About three hundred related to the two domains he chose (the game of three-handed Hearts, and composing canti firmi). The remainder implemented 17 general heuristics for achieving goals. Mostow called these heuristics operationalization methods.

FOO was not designed to solve every possible problem. Its operationalization methods were heuristic, subject to failure or inadequacy. Three examples are:

--If the value of a function must lie in one of several ranges, and all but one of the ranges have been eliminated, then it must lie in the remaining range. (Mostow calls this

the "Pigeonhole Principle.") E.g., if neither you nor your right-hand opponent holds the Queen of Spades, then your left-hand opponent must hold it.*

--Achieve a necessary condition for a goal. This may or may not work. E.g., to take the current trick, play a card that beats the card led. This is necessary but not sufficient, since somebody else may play a card that beats yours.

--Use an untested simplifying assumption. E.g., to avoid taking the Queen of Spades, you may lead an unplayed suit other than spades on the assumption that the other cards in the suit are distributed evenly. Though this assumption sometimes fails, most players would make it routinely.

The power of these methods gives FOO a high order of problem-solving ability. Unfortunately, Mostow did not endow FOO with the knowledge of when to apply which operationalization methods (p. 345 ff.). It was up to the user to tell FOO when to use each method. FOO also could not evaluate its heuristics or modify them in case they failed.

3 The advice-taker/inquirer.

3.1 Initial design.

An advice-taker/inquirer (AT/I) is an advice-taker that seeks help from the Advisor in organizing and interpreting the advice. The concept is due to Findler and was developed within

---

* Since the game is three-handed.

the well-known "Poker Project" at SUNY/AB. The earliest proposal for an AT/I [DIXON 1975] describes it as an interactive tool for Poker System programmers. This proposal was later elaborated but never implemented. Some of its features were implemented in other Poker System projects.

In 1978 S. G. Feuerstein and I took up the AT/I project. Believing that a pure tree structure would be inconvenient for the Advisor, we proposed in its place [SICHERMAN 1978] a collection of linked lists of principles. Each principle would have the form "If <condition> then <action>", where <action> was either a Poker action or a jump to another list. Lists were called sequences. The AT/I would try all the principles of a sequence in order until it found one whose condition held. Thus each sequence was equivalent to a statement of the form "If C_1 then A_1 else if C_2 then A_2 else . . . ". In this way the pattern-identifying component resembled that of a production system with a fixed priority of rules, while the organization was strictly algorithmic. Sequences were not to jump to one another recursively.

The formal syntax of an AT/I strategy was [FINDLER et al. 1979, p. 459]:

```
sequence <=> <label> {[ <principle> ]}+
label <=> {<character>}+
principle <=> {<variable> {<rop> <variable>} {<bop> <variable>
   {<rop> <variable>}+} : <action> {<modifier>}
variable <=> <simple variable> {( <index> )} | <function> |
   <constant>
index <=> <variable>
function <=> <domain-specific function>
simple variable <=> <environmental variable>
constant <=> <integer> | <real> | <boolean constant>
rop <=> ≠ | = | < | > | ≤ | ≥
```

bop <=> AND | OR
action <=> GOTO | <environmental action>
modifier <=> <variable>

• Explanation of notation:
{<item>} means that <item> is optional;
{<item>}+ means that <item> occurs one or more times.

We also announced explicitly what we wanted the AT/I to be able to do [FEUERSTEIN and SICHERMAN 1978]. In doing so we defined "advice" more broadly than usual. The Advisor should be able to issue (A1) definitions, including computational ones; (A2) rules, as described above; (A3) examples, which were like rules but were to be applied by analogy; (A4) inquiries about strategy, meanings of terms, current statistical values, etc.; (A5) orders, to override the strategy. The AT/I should be able to (B1) recognize undefined terms and ask the Advisor to define them; (B2) complain about inconsistent advice; (B3) interpret hypothetical deviations from the current game situation; (B4) illustrate its inquiries by constructing examples and presenting them to the Advisor; (B5) save and retrieve its strategy.

Powers A1 through A4 were to be invoked while the AT/I was not acting in its domain. Power A5 differed from them in being invoked while the AT/I was acting in its domain; instead of refining its strategy, the AT/I was to ignore it temporarily.

To illustrate these capabilities, here is a hypothetical dialogue on the subject of poker:

ADVISOR: If your opponent is a cowboy, bluff often. (A2)
AT/I: What is a cowboy? (B1)
ADV.: A cowboy is a player who usually draws to a four-flush. (A1)
AT/I: You said earlier never to bluff against a beginner. What if the cowboy is a beginner? (B2)
ADV.: Don't bluff. (A2)
AT/I: I can think of another class of situations. For example, suppose you are in first position, above a cowboy, and he has been winning steadily, and he won the previous pot. Your previous advice says don't bluff. (B4)
ADV.: Right. Don't bluff. But suppose you are the dealer in that situation, then you ought to bluff. (A3) Now save your advice on file "X", and retrieve the advice on file "Y". (B5)
AT/I: O.K.
ADV.: Now play 10 games of poker.
. . .
ADV.: Is the opponent on your left a beginner? (A4)
AT/I: Yes, according to your definition.
ADV.: What would you have done if he had raised you?
AT/I: Raised him back. (B3)
ADV.: If he opens the next hand, I want you to call him. (A5)

3.2 Implementation.

By the end of summer 1978 we had a program. It was written partly in CDC assembly language (COMPASS) and partly in FORTRAN IV, using the SLIP package for list structures. The coding of the data structures relied heavily on enumeration constants and bit fields. The program had four parts [FEUERSTEIN 1978]: the interactive communication with the Poker System, which Feuerstein adapted from the existing multi-teletype communication routines

[SICHERMAN 1977]; the *controller*, which supervised the AT/I's submodules; the human interface, or *editor*; and the rule follower, or *executor*. For an overview of the first implementation and early results see [FINDLER *et al.* 1979]. A few details are given here.

The original proposal called for natural-language communication, though with a small vocabulary. Instead we implemented formal-language communication. Later we abandoned the idea of natural language altogether, because it would have been formidable to program (especially in FORTRAN), perhaps not the most efficient form of communication, and not significant for the project. (See also Davis's arguments against natural-language parsing [DAVIS 1979, p. 128].)

The Advisor could define and display sequences but not alter them.

The flow of information and the control structure of the eventual design for the AT/I are shown in Figures 1 and 2.

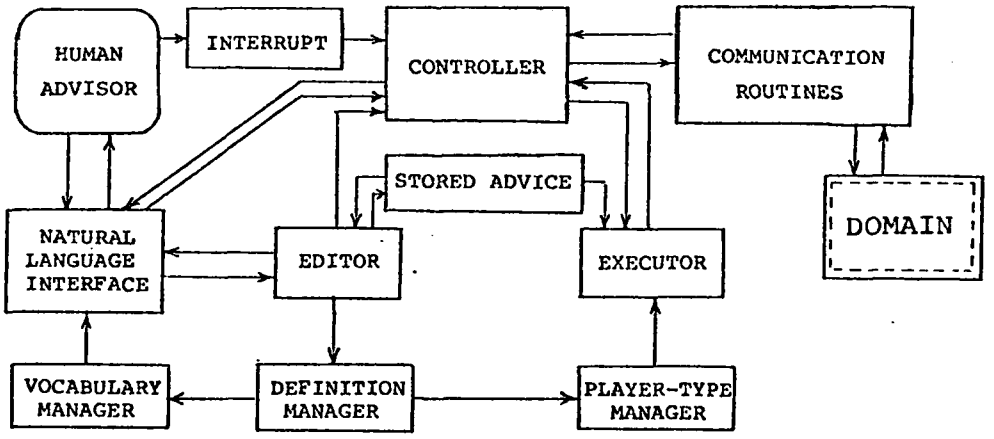


Figure 1. Flow of information in the AT/I.

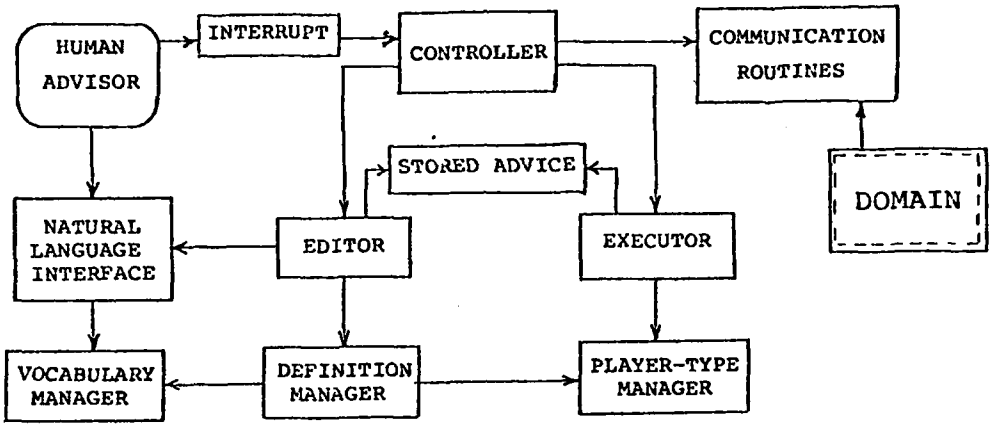


Figure 2. Flow of control in the AT/I.

3.3 Design goals.

We propose to develop an advice-taking system that can

1. ask intelligent questions to resolve apparent contradictions, omissions, or uncertainties in the advice;
2. understand examples, and prepare its own examples to illustrate its strategy;
3. experientialize its advice; i.e., adapt its strategy according to its experience with using it.

The first two capabilities were originally proposed for the Advice-Taker/Inquirer [FINDLER et al. 1979], which has been partially implemented as described above. The third is new. So far as I know, there are no intelligent systems that learn both from an Advisor, as does FOO, and from experience, as does HACKER [SUSSMAN 1974].

I propose also to

(4) develop a mathematical model of advice-taking, with a view to obtaining measures of the power and reliability of advice-taking systems.

3.3.1 Originally proposed work.

The NOS-FORTRAN-SLIP environment does not facilitate interactive programming. If anything, it hinders it. I therefore propose to write the AT/I in a more hospitable language, probably either LISP or a LISP-imbedded language such as ROSIE [GORLIN et al. 1981A; GORLIN et al. 1981B; GORLIN et al. 1981C]. I will adapt as much of the existing FORTRAN code as is worth keeping.

The next step will be to implement as many of the AT/I's intended capabilities as prove feasible. At the least, the AT/I must be able to accept and use definitions of new terms, and the Advisor must be able to describe and rearrange parts of the strategy. Automatic classification of actors, such as was done by the Player-Type System [MORGADO and MARTINS 1978] in the SUNY/AB Poker System, can proably be implemented through the Definition Manager [FINDLER et al. 1979]. It should be routine to give the Advisor broad powers of inquiry.

3.3.2 Domains of application.

The original domain of application for the AT/I was the SUNY/AB Poker System [FINDLER et al. 1972; FINDLER 1977; FINDLER 1978]. For my thesis I intend to use the domain of automatic Air Traffic Control (ATC), whose tasks are complex enough to be

challenging and can be kept simple enough to be manageable. The problem of automating ATC has been studied extensively in the last three or four years [WESSON 1977A; WESSON 1977B; RUCKER 1979; GOLDMUNTZ et al. 1981] and is obviously important in its own right.

A second domain, one unlike ATC, will also be used, mainly to establish that the AT/I is reasonably domain-independent.

3.3.3 Benefits and possible uses.

Some potential benefits of using an AT/I are:

(1) By expressing his strategy formally, the Advisor may come to understand it better.
(2) The AT/I can point out inconsistencies, ambiguities, and other logical problems with the strategy.
(3) The AT/I can find and help to correct practical problems with the strategy.

These observations suggest some possible uses of an AT/I.

Training. An AT/I can be used to train or retrain experts.

Automation. Having absorbed the expert's knowledge, an AT/I can act in his place.

Testing. The AT/I can devise simulations that test a strategy in many ways.

Development. The AT/I can be used in forming, debugging, maintaining, and improving strategies.

Synthesis. An AT/I, working with a system that analyzes several alternative strategies for the same goal and develops a strategy with the best features of each, such as that described in [FINDLER and Van LEEUWEN 1979], could learn from several Advisors and produce a system that outdoes them all.

3.4 Experientialization.

3.4.1 Introduction.

Previous research in automatic advice-taking has focused on problems of operationalization; i.e., of organizing and applying knowledge to accomplish goals. The user must decide how well the resulting agent performs, and what to do if it performs unsatisfactorily. (TEIRESIAS supports this process very well but cannot direct it.)

Let experientialization denote the process by which an advice-taking program refines its advice on the basis of experience. This is a highly significant open problem in AI In an era of limited computer-skilled manpower, the practical value of self-correcting educable systems is inestimable.

To experientialize, the program must determine what to refine and how. For example, it may refine

- rules that fail (by fixing them);
- rules that succeed (by forming more general, more specific, or analogous rules);
- rules that appear to be critical, though it be hard to judge their soundness.

In the process of refining it may

1. ask the Advisor for help;
2. try various changes in the strategy and test them by simulating the environment;
3. form new concepts to replace old ones.

3.4.2 Cognitive processes in experientialization.

To experientialize advice requires three processes: observation, organization, and assimilation. These processes naturally tend to influence one another, but a system is most likely to be powerful if it distinguishes them clearly.*

Observation. The AT/I must be able to observe and record what happens in the environment. To obtain this information it may make its own measurements, or ask the Advisor or a third party. It may have to commit resources to this task, or even devise a strategy for it.

Organization. The AT/I must keep the size of its stored experience low enough to be manageable, and must be able to derive information that it can use to reach its goal. The natural tools for this process are statistical analysis and pattern recognition. Pattern recognition may involve forming new concepts, to supplement those defined by the Advisor.

Assimilation. The AT/I must check its strategy against its experience and take action accordingly. The nature of the actions will depend on whether the environment is cooperative, competitive, or neutral; but their forms can be classified as

---

* See [PERLS et al. 1951] for a discussion of this principle in human cognition.

follows:

Troubleshooting. This means identifying sources of failure and fixing them. This is the meaning of "knowledge refinement" as used by Hayes-Roth et al. [HAYES-ROTH et al. 1980]. Some troubleshooting methods are: consulting the Advisor; simulation; credit assignment; aggression; diplomacy.

Reinforcement. When the strategy succeeds, the AT/I should be encouraged to "appreciate" the causes of its success; i.e., to find new applications and variants of them. As is well known, this is not an easy process to implement. The early history of programmed reinforcement is cratered with failures.

Discovery. This refers to the process of forming new, relevant concepts without immediate regard for the AT/I's goals. It eventually incorporates them or discards them according to some heuristic for measuring relevance.

Desperation. When the AT/I fails, it may do so because of the advice or in spite of it; that is, because the advice is simply inadequate for the task. The AT/I should recognize when it is failing in spite of the advice and, perhaps with the Advisor's help, develop radically new concepts and techniques to deal with the problem.

3.4.3 Methods of experientialization.

If the AT/I's strategy is composed of rules of the form "to accomplish A in situation B, do C", then it has an obvious way to compare its strategy with its experience: just count how often

each rule succeeds and fails. The results can be applied immediately to credit assignment. Perhaps because of its lack of glamor, this statistical method, while common in other areas of computer science, is rare in artificial intelligence.

This method may not be sufficient. The AT/I may not be able to observe whether B holds, or A is indeed accomplished, or even whether it succeeded in doing C. Even if the method were sufficient, it would hardly solve the problem of experientialization. The AT/I needs a battery of heuristics, somewhat like FOO's battery of operationalization heuristics, to transform its experience into action. I cannot say what these heuristics will be like. They will have to be discovered by experiment. Most of them will involve cooperation between the Advisor and the AT/I.

3.4.4 Testing by experientialization.

If an AT/I is to be used in the field, it must be tested thoroughly. Ideally it should encounter every possible situation, so as to discover every possible failure. In practice this can seldom be done. It is a sad truism in systems programming that complex systems cannot be completely debugged. The main reason is that the possible situations are too numerous to generate artificially, and reality generates them with widely varying frequencies.

But it is possible to generate a large number of different situations to test the AT/I. This will not ensure that the AT/I's strategy is perfectly sound, but it may verify that the strategy is sound enough for all practical purposes.

3.5 Mathematical models of advice.

Define an interactive environment as a system of equations

$$y_i = f_i(t, x_1, \ldots, x_m, y_1, \ldots, y_n, z_1, \ldots, z_p) \quad (1 \leq i \leq n);$$
$$z_i = g_i(t, x_1, \ldots, x_m, y_1, \ldots, y_n, z_1, \ldots, z_p) \quad (1 \leq i \leq p).$$

The variable t represents time. The values of the variables $x_1, \ldots, x_m$ are set by the user. The functions f_i and g_i describe how the environment behaves; they may express complicated logical conditions as well as numerical functions. The variables $y_1, \ldots, y_n$ are measurable; the variables $z_1, \ldots, z_p$ are not.

Partial measurability can be expressed in this model. For example, suppose that z_k takes values $0.0 \leq z_k \leq 1.0$ and is observable only when $z_k < 0.5$. The corresponding measurable variable y_j is defined by

$$y_j = \begin{cases} z_k & \text{if } z_k < 0.5; \\ \text{"unknown"} & \text{otherwise.} \end{cases}$$

Likewise, loss of precision and loss of accuracy can be expressed by equations like

$$y_j = \begin{cases} 0.0 & \text{if } 0 \leq z_k < 0.05; \\ 0.1 & \text{if } 0.05 \leq z_k < 0.10; \\ \quad . \quad . \quad . \quad . & \\ 1.0 & \text{if } 0.95 \leq z_k \leq 1.00 \end{cases}$$

and

$$y_j = z_k + cz_e$$

where z_e represents a source of error.

This model does not account for events that are random; i.e., whose values cannot be determined from other quantities.* We can extend the model to allow random events by simply allowing the functions f_i and g_i to take random variates as arguments:

$$y_i = f_i(t,x1,\ldots,xm,y1,\ldots,yn,z1,\ldots,zp,r) \quad (1 \leq i \leq n);$$
$$z_i = g_i(t,x1,\ldots,xm,y1,\ldots,yn,z1,\ldots,zp,r) \quad (1 \leq i \leq p).$$

We can view the x_i as being inputs to the environment and the y_i as outputs. An actor is an entity that seeks to control the y_i by regulating the x_i.

Define a pure interactive strategy as a system of functions

$$w_i = d_i(t,w1,\ldots,wl,x1,\ldots,xm,y1,\ldots,yn) \quad (1 \leq i \leq l);$$
$$x_i = e_i(t,w1,\ldots,wl,x1,\ldots,xm,y1,\ldots,yn) \quad (1 \leq i \leq m).$$

The w_i are the actor's internal variables, the x_i are his actions in the environment, and the y_i are his observations. Define a mixed interactive strategy by allowing random variates for arguments:

$$w_i = d_i(t,w1,\ldots,wl,x1,\ldots,xm,y1,\ldots,yn,r) \quad (1 \leq i \leq l);$$
$$x_i = e_i(t,w1,\ldots,wl,x1,\ldots,xm,y1,\ldots,yn,r) \quad (1 \leq i \leq m).$$

A strategy and an environment are symmetrical, as can be seen from their forms. Together they form a closed system.

The analytic control problem is: given an environment (f,g), to find a strategy (d,e) that will maximize a given function $\phi(t,y)$.

---

* This definition is necessarily imprecise because randomness is not an absolute concept. It can be joint or relative, as in the Heisenberg Uncertainty Principle.

If δ is boolean-valued, this amounts to achieving a condition. The presence of the variable t means that the problem may be to maximize a function or achieve a condition over time.

This formulation of the problem assumes that the functions f_i and g_i are known. In practice they are often known imperfectly or not at all; and even if their behavior is familiar, it may be too complex to be described completely.* We therefore assume that the actor's knowledge of the environment is based on experience. This experience need not be the actor's own. It may be the experience of one or many others, communicated by an Advisor.

Of course, not all that can be communicated is experience. The properties of a communication that are relevant in modeling advice-taking are: immediacy, noise, and community of interest.

By immediacy I mean the freshness of the Advisor's knowledge. This is affected by storage error (i.e., deterioration) and directness of experience. The Advisor is more reliable when relating what he saw than when repeating what he was told.

Transmission noise includes the Advisor's mistakes in describing his knowledge, translating it, expressing it, and the system's mistakes in receiving it, translating it, and interpreting it.

Community of interest means that the goal towards which the system intends to apply its knowledge is also a goal of the Advisor. When this is not so, the communication may contain unreliable or misleading information.

---

* An example is the rules for resolving ko in the game of go.

4 References

[BIERMANN 1976] Biermann, A. W., "Approaches to automatic programming." In Advances in Computers, v. 15 (1976), pp. 1-63.

[DAVIS 1979] Davis, R., "Interactive transfer of expertise: acquisition of new inference rules." Artificial Intelligence, v. 12 (1979), pp. 121-157.

[DIXON 1975] Dixon, R., "The interactive advice taking program." Memo. no. 36, SUNY/AB Poker Group (1975).

[FEUERSTEIN 1978] Feuerstein, S. G., "The AT/I system." Memo. no. 174, SUNY/AB Poker Group (1978).

[FEUERSTEIN and SICHERMAN 1978] Feuerstein, S. G., and Sicherman, G. L., "Work proposed for the summer on the AT/I." Memo. no. 166, SUNY/AB Poker Group (1978). Memo. no. 102, SUNY/AB Poker Group (1976).

[FINDLER 1977] Findler, N. V., "Studies in machine cognition using the game of Poker." Comm. ACM, v. 20 (1977), pp. 230-245.

[FINDLER 1978] Findler, N. V., "Computer poker." Scientific American, v. 239 (1978), pp. 144-151.

[FINDLER et al. 1972] Findler, N. V., Klein, H., Gould, W., Kowal, A., and Menig, J., "Studies on decision making using the game of Poker." In Proc. IFIP Congress 1971 (1972), pp. 1448-1459.

[FINDLER and Van LEEUWEN 1979] Findler, N. V., and van Leeuwen, J., "On the complexity of decision trees, the quasi-optimizer, and the power of heuristic rules." Information and control, v. 40 (1979), pp. 1-19.

[FINDLER et al. 1979] Findler, N. V., Sicherman, G. L., and Feuerstein, S. G., "Teaching strategies to an Advice-Taker/Inquirer System." Proc. Euro IFIP 79 (1979), pp. 457-465.

[GOLDMUNTZ et al. 1981] Goldmuntz, L., Kefaliotis, J. T., Weathers, D., Kleiman, L. A., Rucker, R. A., and Schuchman, L., "The AERA concept." 1981.

[GORLIN et al. 1981A] Gorlin, D. M. et al., "Programming in ROSIE: an introduction by means of example." Report N-1646, Rand Corp., 1981.

[GORLIN et al. 1981B] Gorlin, D. M. et al., "The ROSIE language reference manual." Report N-1647, Rand Corp., 1981.

[GORLIN et al. 1981C] Gorlin, D. M. et al., "Rationale and motivation for ROSIE." Report N-1648, Rand Corp., 1981.

[HAYES-ROTH et al. 1980] Hayes-Roth, F., Klahr, P., and Mostow, D. J., "Knowledge acquisition, knowledge programming, and knowledge refinement." Report R-2540, Rand Corp., 1980.

[McCARTHY 1968] McCarthy, J., "Programs with common sense." In Minsky, M. (ed.), Semantic Information Processing (1968), pp. 403-417.

[MORGADO and MARTINS 1978] Morgado, E. J. M., and Martins, J. E. P., "User-defined player types." In Prieur, J. R., Morgado, E. J. M., Martins, J. E. P., and Findler, N. V., Users' Guide

to the Poker System. Volume III: Manual on the "New" Tournament Mode. 2nd ed. Tech. manual no. 5, SUNY/AB Poker Group, 1978.

[MOSTOW 1981] Mostow, D. J., "Mechanical transformation of task heuristics into operational procedures." Ph.D. thesis, Carnegie-Mellon Univ., 1981.

[PERLS et al. 1951] Perls, F. S., Hefferline, R. F., and Goodman, P., Gestalt Therapy: Excitement and Growth in the Human Personality. N.Y., Julian, 1951.

[RUCKER 1979] Rucker, R. A., "Automated en route ATC (AERA): Operational concepts, Package 1 description, and issues." 1979.

[SHORTLIFFE and BUCHANAN 1975] Shortliffe, E. H., and Buchanan, B. G., "A model of inexact reasoning in medicine." Mathematical Biosciences, v. 23 (1975), pp. 351-379.

[SICHERMAN 1977] Sicherman, G. L., "New communication routines: level 1 documentation." In Menig, J., Findler, N. V. et al., Users' Guide to the Poker System, 3rd ed., Tech. manual no. 1, SUNY/AB Poker Group, 1978, App. E.

[SICHERMAN 1978] Sicherman, G. L., "Progress in the advice-taker/inquirer design." Memo. no. 159, SUNY/AB Poker Group (1978).

[SUMMERS 1975] Summers, P. D., "A methodology for LISP program construction from examples." Proc. 3rd ACM Symp. Prin. Prog. Lang. (1975), p. 68.

[SUSSMAN 1974] Sussman, G. J., A Computer Model of Skill Acqui-

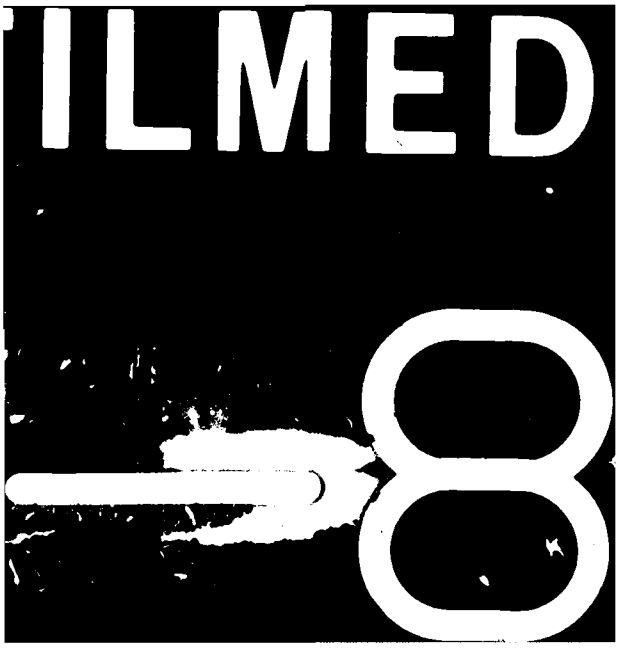
sition. N.Y., Amer. Elsevier, 1974.

[WESSON 1977A] Wesson, R. B., "Planning in the world of the air traffic controller." Proc. 5th Intern. Jt. Conf. Artif. Intell., 1977.

[WESSON 1977B] Wesson, R. B., "Problem-solving with simulation in the world of an air traffic controller." Ph.D. thesis, Univ. of Texas at Austin, 1977.

[WINOGRAD 1972] Winograd, T., Understanding Natural Language. Academic Press, 1972.

[ZOBRIST and CARLSON 1973] Zobrist, A. L., and Carlson, F. R. Jr., "An advice-taking chess computer." Scientific American, v. 228 (1973), pp. 92-105.


ILMED
8